REMARKS

Applicants request that the attorney docket number of this case be changed to NC 84,888.

Claims 1-10 and 12-30 are pending in the application. Claim 11 has been canceled by this amendment without prejudice. No claims are presently allowed.

Claims 15, 17, 19, and 22 have been amended to change "determining that" or "determining" to "determining whether."

Claims 19 and 22 have been amended to change "the first instruction identifies a second thread" to "a second thread depends on said first instruction."

Claim 22 has been amended to add "in said counter," in reference to a non-zero value.

Claim 23 has been amended to add a missing "and."

Claim Objections

Claim 11 was objected to for reciting "is" instead of "are." The claim has been canceled by this amendment.

Claims 15, 17, 19, and 22 were objected to for reciting "determining that." This has been changed to "determining whether," as suggested by the Examiner.

Claim Rejections – 35 U.S.C. § 112

Claim 11 has been rejected under 35 U.S.C. § 112, second paragraph as indefinite for reciting elements as "said" without antecedent basis. This claim has been canceled.

Claims 19 and 22 have been rejected under 35 U.S.C. § 112, second paragraph as indefinite for reciting "first instruction identifies a second thread." This has been changed to "second thread depends on said first instruction," as suggested by the Examiner.

Claim 22 has been rejected for under 35 U.S.C. § 112, second paragraph as indefinite for reciting "a non-zero value." This has been changed to "a non-zero value in said counter," similar to the Examiner's suggestion.

Claim Rejections – 35 U.S.C. § 102

Claims 1-14 and 23 have been rejected under 35 U.S.C. § 102(a) as being anticipated by

Akkary et al., US 6,182,210.

Akkary discloses a method and apparatus for processing dependent threads using predictive methods.

Claim 1 of the present application is to an apparatus for instruction-level parallelism in a processing element, comprising: an instruction control unit; a first instruction buffer; a second instruction buffer; a dependency counter; an execution switch; and an execution unit.

The Examiner pointed to the data and dependency array (DAD) 206A of Akkary as anticipating the dependency counter of claim 1. The dependency counter is incremented upon the execution of a first instruction in a first thread on which a second instruction in a second thread depends (paragraph 0008). Execution of the second instruction will depend on whether the counter has reached a threshold value (0023). This way, each dependent instruction will not be executed until the proper number of dependee instructions have already been executed. In one embodiment, each thread has a group of dependency counters, one for each other thread (0055). Thus, there is a counter for every possible dependency relationship among the threads.

The DAD array of Akkary is shown in detail in Fig. 13 and described at col. 10, line 58-col. 11, line 34. It contains a replay count field for each instruction in a thread. The replay count of an instruction is incremented every time that same instruction is replayed in the pipeline (col. 11, lines 9-12). This has no relation to the execution of a different instruction from a different thread as in the present claim 1. The DAD array also has a group of dependency fields for each instruction in the thread. However, these single-bit flags show relationships between registers and instructions in the thread (col. 11, lines 18-27), not between instructions in different threads.

Claims 2-6 depend from and contain all the limitations of claim 1 and are asserted to distinguish from the reference in the same manner as claim 1.

Claim 7 of the present application is to an apparatus for processing instructions in multiple threads comprising an instruction buffer, a dependency counter, an instruction control unit, and an execution switch. The instruction control unit detects instruction dependency bits and increments and decrements the dependency counter.

The Examiner pointed to the replay count of Akkary as anticipating incrementing and decrementing the dependency counter of claim 7. The dependency counter is incremented upon the execution of a first instruction in a first thread on which a second instruction in a second thread depends. As explained above, the replay count of an instruction is incremented every

time that same instruction is replayed in the pipeline. This has no relation to the execution of a different instruction from a different thread as in the present claim 7.

Claims 8-10 depend from and contain all the limitations of claim 7 and are asserted to distinguish from the reference in the same manner as claim 7.

Claim 12 is to an apparatus for instruction-level parallelism, comprising an instruction buffer and an instruction control unit. The instruction buffer holds first and second instructions associated with first and second threads. The instruction control unit detects instruction dependency bits that indicate dependency between an instruction and one or more threads other than the thread with which the instruction is associated.

The Examiner pointed to the instruction queue array 202A of Akkary as anticipating the instruction buffer of claim 12. The instruction buffer simultaneously holds both the first and second instructions, being from different threads. However, the instruction queue array can hold instructions from only one thread at a time. "Instruction queue array 202A receives instructions ... that are part of a particular thread. ... Instructions that are part of another thread are written into an instruction queue array of a different trace buffer or into instruction queue array 202A at a different time." (Col. 9, lines 46-52.) Thus, a given instruction queue array of Akkary never holds instructions from two different threads at the same time, as recited in the current claim 12.

The Examiner also pointed to the dependency field of Akkary as anticipating detecting dependency bits that indicate dependency between an instruction and one or more threads other than the thread with which the instruction is associated. However, the dependency field indicates which registers are inputs to each instruction, whether directly or through previous instructions in the trace. This information is used when a value is misspeculated to determine which instructions need to be replayed (col. 11, lines 18-27). The dependency field indicates relationships between registers and instructions, not between instructions of different threads.

Claims 13 and 14 depend from and contain all the limitations of claim 12 and are asserted to distinguish from the reference in the same manner as claim 12.

Claim 23 is to method for processing instructions in multiple threads, comprising: loading a first instruction associated with a first thread; and detecting dependency between the first instruction and a second instruction associated with a second thread based on dependency bits in an instruction buffer and the value of a dependency counter.

The Examiner pointed to the dependency field of Akkary as anticipating the dependency bits. As explained above the dependency field is different from the dependency bits.

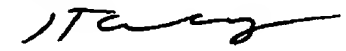
The Examiner also pointed to the value or PRID as anticipating the value of the dependency counter. The value of the dependency counter keeps track of dependency relationships. The value or PRID is the result of a calculation by a single instruction or a reference to a register containing such a value (col. 11, lines 1-3). This is not related to dependencies between instructions.

The Examiner also pointed to the DAD of Akkary as anticipating the dependency counter. As explained above, these components are not equivalent.

In view of the foregoing, it is submitted that the application is now in condition for allowance.

In the event that a fee is required, please charge the fee to Deposit Account No. 50-0281, and in the event that there is a credit due, please credit Deposit Account No. 50-0281.

Respectfully submitted,



Joseph T. Grunkemeyer
Reg. No. 46,746
Phone No. 202-404-1556
Office of the Associate Counsel
(Patents), Code 1008.2
Naval Research Laboratory
4555 Overlook Ave, SW
Washington, DC 20375-5325